Discovering Band Order Dependencies

Li, Pei ; Szlichta, Jaroslaw ; Böhlen, Michael ; Srivastava, Divesh

Abstract: We introduce band ODs to model the semantics of attributes that are monotonically related with small variations without there being an intrinsic violation of semantics. To make band ODs relevant to real-world applications, we make them less strict to hold approximately with some exceptions. Since formulating integrity constraints manually is cumbersome, we study the problem of automatic approximate band OD discovery. We devise an algorithm that determines the optimal solution in polynomial time. We perform a thorough experimental evaluation of our techniques over real-world and synthetic datasets.

DOI: https://doi.org/10.1109/ICDE48307.2020.00193

Discovering Band Order Dependencies

Pei Li [▽1], Jaroslaw Szlichta [#2], Michael Böhlen [$3], Divesh Srivastava [÷4]

[▽] Onedot A.G., Switzerland, [#] Ontario Tech University, Canada,
[$] University of Zürich, Switzerland, [÷] AT&T Labs-Research, United States
[1] pei.li@onedot.com, [2] jarek@uoit.ca, [3] boehlen@ifi.uzh.ch, [4] divesh@research.att.com

***Abstract*—We introduce band ODs to model the semantics of attributes that are monotonically related with small variations without there being an intrinsic violation of semantics. To make band ODs relevant to real-world applications, we make them less strict to hold *approximately* with some exceptions. Since formulating integrity constraints manually is cumbersome, we study the problem of automatic approximate band OD discovery. We devise an algorithm that determines the optimal solution in polynomial time. We perform a thorough experimental evaluation of our techniques over real-world and synthetic datasets.**

I. Introduction

Modern data-intensive applications critically rely on high quality data to ensure that analyses are meaningful and do not fall prey to the garbage in, garbage out (GIGO) syndrome. In constraint-based data cleaning, dependencies are used to specify data quality requirements. Data that are inconsistent wrt the dependencies are identified as erroneous, and modifications to the data are performed to re-align the data with the dependencies. Previous work has focused on functional dependencies (FDs) [1]. Several extensions to the notion of an FD have been studied, including *order dependencies* (ODs) [2], [3], [4], [5], [6], [7], [8], [9], which express rules involving order. We introduce a novel data dependency: *approximate band OD*. *Band ODs* express order relationships between attributes with small variations causing traditional functional dependencies and rules involving order including ODs [3], [5], [6], sequential dependencies [2] and denial constraints [10] to be violated without actual violation of semantics. To match real world scenarios, we allow band ODs to hold *approximately* with some exceptions.

Table I contains 9 sample releases of the *Music* dataset from Discogs[1]. When lexicographically ordered by attribute catalog number cat#, the release date (encoded using attributes year and month) is also approximately ordered.

Note that tuple t_3 has a smaller cat# than t_4 (CDW46012 $<$ CDW46046), but is released a few months later than tuple t_4 (1996/Feb $>$ 1995/Oct). This is common in the music industry as cat# is often assigned to a record before it is actually released at the production stage. Thus, tuples with delayed release dates will slightly violate an OD between cat# and (year, month). A permissible range to accommodate these small variations is called a *band*. Attribute year has also an erroneous value (tuple t_2).

Data dependencies to identify data quality errors can be obtained manually through a consultation with domain experts, but this is known to be an expensive, time consuming, and error-prone process [1], [2], [3], [5]. Thus, automatic approaches to discover data dependencies to identify data quality issues are needed. The key technical problem that we study is how to automatically discover approximate band order dependencies.

TABLE I
MUSIC RECORDS.

id	release	country	year	month	cat#
t_1	Unplugged	Canada	1992	Aug	CDW45024
t_2	Mirror Ball	Canada	**2012**	Jun	CDW45934
t_3	Ether	Canada	1996	Feb	CDW46012
t_4	Insomniac	Canada	1995	Oct	CDW46046
t_5	Summerteeth	Canada	1999	Mar	CDW47282
t_6	Sonic Jihad	Canada	2000	Jul	CDW47383
t_7	Title of...	Canada	1999	Jul	CDW47388
t_8	Reptile	Canada	2001	Mar	CDW47966
t_9	Always...	Canada	2002	Feb	CDW48016

We make the following contributions in this paper.

- We define a novel *band* OD integrity constraint based on small variations causing traditional ODs to be violated without an actual violation of application semantics. To make band ODs applicable to real-world data, we relax their requirements to hold *approximately*.
- We formulate the *approximate band OD discovery problem*. We devise an algorithm that finds the optimal solution in polynomial time based on the proposed notion of *longest monotonic bands* (LMBs) to identity longest subsequences of tuples that satisfy a band OD.
- We experimentally demonstrate the effectiveness of our solution, and compare our techniques with baseline methods on real-world and synthetic datasets.

II. Preliminaries

Letter r denotes a relation. Italic letters A, B and C denote single attributes. Also, s and t denote tuples in r and $s.A$ denotes the value of an attribute A in tuple s. $dom(A)$ denotes the domain of attribute A. Bold letters **X, Y** and **Z** denote lists of attributes. $[A,B,C]$ denotes an explicit list of attributes. $dom(\mathbf{X}) = dom(A) \cdot dom(B) \cdot dom(C)$ denotes the domain of $\mathbf{X}$, where $\mathbf{X} = [A,B,C]$. $s.\mathbf{X}$ denotes the value of the list of attributes $\mathbf{X}$ in tuple s. Let $d : dom(\mathbf{X}) \cdot dom(\mathbf{X}) \rightarrow \mathbb{R}$ be a *distance function* defined on the domain of $\mathbf{X}$. Distance function d satisfies *anti-symmetry, triangle inequality* and *identity of indiscernibles* properties. We consider $d(x_1,x_2) = ||x_2|| - ||x_1||$, where $||x||$ denotes the norm of the value list x.

[1] www.discogs.com

Definition 2.1: Given $\mathbf{Y}$ over a relation r, let Δ be a constant value. For two tuples $t,s \in r$, $t \preceq_{\Delta,\mathbf{Y}} s$ if $d(s.\mathbf{Y},t.\mathbf{Y}) \leq \Delta$. Let $t \preceq_{\mathbf{Y}} s$ be the operator $t \preceq_{\Delta,\mathbf{Y}} s$, where $\Delta = 0$. □

Definition 2.2: Given a band-width Δ, lists of attributes $\mathbf{X}$ and $\mathbf{Y}$ over a relation r, a *band order dependency* (band OD) denoted by $\mathbf{X} \mapsto_{\Delta} \mathbf{Y}$ holds over a table r if $t \preceq_{\mathbf{X}} s$ implies $t \preceq_{\Delta,\mathbf{Y}} s$ for every tuple pair $t,s \in r$. □

Band ODs specify that when tuples are ordered increasingly on antecedent (left-hand-side), their consequent (right-hand-side) must be ordered non-decreasingly within the specified band-width.

Example 2.3: A band OD $\mathsf{cat\#} \mapsto_{\Delta=1} \mathsf{year}$ holds over tuples $\{t_1, t_3\text{–}t_9\}$ in Table I with a band-width of one year. A band OD $\mathsf{cat\#} \mapsto_{\Delta=12} [\mathsf{year}, \mathsf{month}]$ holds over tuples $\{t_1, t_3 - t_9\}$ in Table I with a band-width of 12 months. □

In real-world applications, band ODs often hold *approximately* with some exceptions. We formally define the problem of approximate band OD discovery in Section III.

III. Band OD Discovery

A. Longest Monotonic Band

To discover approximate band order dependencies, we introduce the notion of a *longest monotonic band* (LMB) to identify the longest subsequences of tuples that satisfy a band OD. In contrast to previous methods [2], [3], [5], LMBs allow for *slight variations*. We define LMBs with respect to a band OD $\mathbf{X} \mapsto_{\Delta} \mathbf{Y}$. In the remaining, $T = \{t_1, t_2, \cdots, t_n\}$ denotes a sequence of tuples ordered lexicographically by $\mathbf{X}$ in ascending order.

Definition 3.1 (Longest Monotonic Band): Given a sequence of tuples $T = \{t_1, t_2, \cdots, t_n\}$, list of attributes $\mathbf{Y}$ and band-width Δ, a *monotonic band* (MB) is a subsequence of tuples $M = \{t_i, \cdots, t_j\}$ over T, such that $\forall_{k_1,k_2 \in \{i,\cdots,j\}, k_1 < k_2}\ t_{k_1} \preceq_{\Delta,\mathbf{Y}} t_{k_2}$. The longest subsequence M satisfying this condition over T is called a *longest monotonic band* (LMB). □

Example 3.2: Consider band OD $\mathsf{cat\#} \mapsto_{\Delta=1} \mathsf{year}$ over Table I ordered by an attribute cat#. Note that LMBs are *not* necessarily contiguous subsequences of tuples. For example, in Table I a LMB over a sequence of tuples T = $\{t_1\text{–}t_9\}$ includes tuples $t_1 \cup \{t_3\text{–}t_9\}$. □

Definition 3.3 (Maximal Tuple): Given a sequence of tuples $T = \{t_1, \cdots, t_n\}$ and a list of attributes $\mathbf{Y}$, a tuple $t_i \in T$ is a *maximal tuple*, denoted as $\max_{\mathbf{Y}}(t_1, \cdots, t_n)$, if $\forall_{j \in \{1,\cdots,n\}}$ $d(t_j.\mathbf{Y}, t_i.\mathbf{Y}) \geq 0$. □

Example 3.4: Given a sequence of tuples $T = \{t_3\text{–}t_7\}$ over Table I, the tuple t_6 is a maximal tuple with respect to the attribute year. □

We compute a LMB in sequence T by reducing the problem to finding LMBs in sub-sequences of T. Once LMBs in sub-sequences of T are enumerated, the longest one is picked as a LMB.

Let $T[i]$ denote the prefix of a sequence of tuples T of length i, i.e., $T[i] = \{t_i, t_2, \cdots, t_i\}$, where $i \in [1,n]$. Among all MBs that end at t_i in $T[i]$, let MB_i be the one of longest length $l(i)$, where m_i is the maximal tuple in MB_i. The solution of $l(i), i \in [1,n]$ has an *optimal substructure* property.

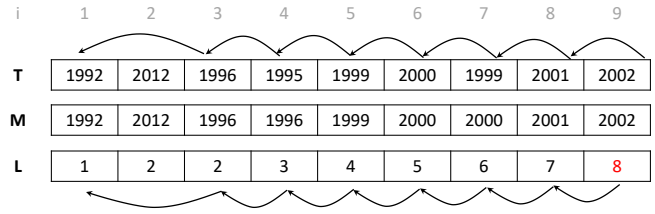

Fig. 1. Compute LMB in Table I, Δ=1; tuples in array M are represented by year.

$$l(i) = \begin{cases} \max_{j \in \{1,\cdots,i-1\} \text{ and } m_j \preceq_{\Delta,\mathbf{Y}} t_i} \{l(j)+1\} & i > 1 \\ 1 & \textit{otherwise} \end{cases} \quad (1)$$

Algorithm 1 describes how to compute a LMB in T (illustrated by Example 3.5 and Example 3.6 below). Let L be an array of length n, where element $L[i], i \in [1,n]$ stores the longest length of MBs ending at t_i in sub-sequence $T[i]$, and M be an array of length n, where element $M[i], i \in [1,n]$ stores the maximal tuple m_i of MB_i.

Equation 1 suggests that in order to find a LMB in sequence $T[i]$, given those of $T[1]$ till $T[i-1]$, we need to check whether tuple t_i can extend the length of any existing MBs in $T[i-1]$. If yes, we only need to keep the one with the longest length (Line 10 in Algorithm 1); otherwise, t_i forms a singleton MB, and its length may be potentially extended by tuples in $T[i+1, \cdots, n]$.

Algorithm 1: Computing LMB

```
   input  : T = {t_1, t_2, ..., t_n}, band width Δ
   output : LMB in T
1  for i ← 1 to n do
2  |  L[i] ← 1; M[i] ← t_i
3  k = 1
4  for i ← 2 to n do
5  |  for j ← 1 to i − 1 do
6  |  |  m_j ← M[j]
7  |  |  if m_j ⪯_{Δ,Y} t_i and L[i] ≤ L[j] + 1 then
8  |  |  |  L[i] ← L[j] + 1
9  |  |  |  M[i] ← max_Y(m_j, t_i)
10 |  |  |  k ← max{L[i], k}
11 for i ← n to 1 do
12 |  if L[i] = k then
13 |  |  LMB[k] ← t_i
14 |  |  k ← k − 1
15 return LMB
```

Example 3.5: Assume $T = \{t_1 - t_9\}$ over Table I, $\mathbf{Y} = [\mathsf{year}]$ and $\Delta = 1$. We consider the computation of an LMB in sequence T. Initially, $L[i] = 1$ and $M[i] = t_i$ for each $i \in \{1, \cdots, 9\}$.

We first consider t_2 with year 2012. Since $M[1] = t_1 \preceq_{\Delta,\mathbf{Y}} t_2$, t_2 can extend MB_1, we get $\mathsf{MB}_2 = \{t_1, t_2\}$ of length 2 with maximal tuple t_2. Therefore, we set $L[2] = 2$ and $M[2] = \max_{\mathbf{Y}}(M[1], t_2) = t_2$. Similarly, we set $L[3] = 2$ and $M[2] = t_3$ as t_3 can extend MB_1.

We next consider t_4 with year 1995. We check if t_4 can extend $\mathsf{MB}_1, \mathsf{MB}_2$ and MB_3 by examining their maximal tuples, and find both $M[1] = t_1 \preceq_{\Delta,\mathbf{Y}} t_4$ and $M[3] = t_3 \preceq_{\Delta,\mathbf{Y}} t_4$. Since MB_1 has length 1 ($L[1] = 1$) and MB_3 has length 2 ($L[3] = 2$), we let t_4 extend MB_3 and set $L[4] = 3$ and $M[4] = \max_{\mathbf{Y}}(M[3], t_4) = t_3$. The remaining tuples are processed accordingly, reported in Figure 1. □

Once array L has been computed, we know that its maximal value is the length of a LMB in T. Next, we describe how to compute a LMB over T given the length of MB_i stored in L. Let k be the largest value in array L, i.e., there exists a LMB of length k in T.

We scan array L in reverse order to construct the path of a LMB (Line 11 in Algorithm 1). Once we find $L[i_k] = k$, then t_{i_k} is the last tuple in the LMB (Line 13 in Algorithm 1). We continue scanning L until we find the first $k-1$ in $L[i_{k-1}]$. Then, $t_{i_{k-1}}$ is found as the $k-1^{th}$ tuple in the LMB. We keep scanning L until all k tuples in the LMB are found.

Example 3.6: Consider $T = \{t_1 - t_9\}$ over Table I, $\mathbf{Y} = [\mathsf{year}]$ and $\Delta = 1$. Figure 1 shows the array L for finding a LMB.

To find a LMB, L is scanned to find the largest value 8 in $L[9]$. Thus, a LMB with length 8 exists in T and t_9 is its 8^{th} tuple. By a reverse scan (marked with arrows in Figure 1) from $L[9]$, the 7^{th} tuple t_8 is found. The operation is continued until all tuples in a LMB are found; i.e., $\{t_1('92), t_3('96), t_4('95), t_5('99), t_6('00), t_7('99), t_8('01), t_9('02)\}$. □

Theorem 1: Alg. 1 correctly finds a LMB in a sequence of tuples T of size n in $O(n^2)$ time and $O(n)$ space. □

Proof. To find a LMB in the sequence of tuples T, the key is to find the length of a LMB by using Equation 1. We consider all the different cases in Equation 1.

If $i = 1$, there is only one element t_i in sequence T that belongs to a LMB of length 1. If $i > 1$, assume MB_i is the MB of the longest length among all MBs ending at t_i in sub-sequence $T[i]$. Let MB_j, $j < i$ be the sub-sequence that contains all elements but t_i in MB_i, i.e., $\mathsf{MB}_i = \mathsf{MB}_j \cup \{t_i\}$. If MB_j is empty, then $\mathsf{MB}_i = \{t_i\}$ has length one ($l(i) = 1$). Otherwise, MB_j is not empty, and its maximal tuple is m_j. Since $m_j \preceq_{\Delta,\mathbf{Y}} t_i$, t_i can extend the band MB_j by length 1. Therefore, $l(i) = \max_{j<i, \wedge m_j \preceq_{\Delta,\mathbf{Y}} t_i}\{l(j)+1\}$.

We next prove that Algorithm 1 returns the length of a LMB in sequence T. Algorithm 1 evaluates $l(i)$ by storing the value $l(i)$ in array L. As i increases from 1 to n, the lengths of LMBs in sub-sequence $T[1]$ till $T[n]$ are all computed. Algorithm 1 returns the maximal value in array L, which corresponds to the length of the LMB in T.

For each tuple t_i in the sequence T of length n, it takes time $O(n)$ to update arrays M and L. Therefore, it takes time $O(n^2)$ to find a LMB in the sequence T. Each tuple t_i inserts one value into an array L of length n; thus, Algorithm 1 takes space $O(n)$. □

B. Discovery Problem

In practice, band ODs may not hold exactly, due to errors in the data. We thus define approximate band ODs that hold with some exceptions. Given a band OD $\mathbf{X} \mapsto_\Delta \mathbf{Y}$, where T is a sequence of tuples ordered by $\mathbf{X}$, our goal is to verify whether a band OD holds, such that inconsistent tuples that severely violate a band OD are few.

As in prior work on functional dependency discovery [1], we compute the minimum number of tuples than must be removed from the given sequence T for the band OD to hold. We define the problem of discovering approximate band ODs as follows.

Definition 3.7: Let $\mathbf{X} \mapsto_\Delta \mathbf{Y}$ be a band OD, T be a sequence of tuples over a table t, ordered by $\mathbf{X}$. Given a threshold ε, $0 \le \varepsilon \le 1$, the problem of discovering approximate band ODs is to find band ODs φ, such that $e(\varphi) \le \varepsilon$, where $e(\varphi) = min(\{|r| \mid r \subseteq t, t \setminus r \models \varphi\})/|t|$. □

We call band ODs that hold approximately with some exceptions *approximate band ODs* (abODs). The *approximate band ODs* discovery problem can be solved in quadratic time by finding a LMB in T. Therefore, the minimal set of tuples that violate a band OD $\mathbf{X} \mapsto_\Delta \mathbf{Y}$ are the set s of *inconsistencies*, each of which satisfies $s \in T, s \notin \mathsf{LMB}$.

Lemma 3.8: The problem of discovering approximate band ODs is solvable by finding a LMB with approximation ratio $|s \notin \mathsf{LMB}| \;/\; |s \in \mathsf{T}|$.

Proof. The proof follows directly from the definition of LMBs (Definition 3.1). □

Example 3.9: Consider the approximate band OD $\mathsf{cat\#} \mapsto_{\Delta=1} \mathsf{year}$ over table t in Table I with approximation ratio ε = 15%. Given Example 3.6, we found a LMB of length 8, i.e., $\{t_1('92), t_3('96), t_4('95), t_5('99), t_6('00), t_7('99), t_8('01), t_9('02)\}$. Since $8/9 = 0.89 > 0.85$, the approximate band OD is satisfied over table t.

TABLE II
DISCOVERY QUALITY ON *Music* AND *Car* DATASETS.

	LMS			LMB		
	F-1	Precision	Recall	F-1	Precision	Recall
Music-Real	0.61	0.43	0.99	0.97	0.94	0.99
Car-Real	0.90	0.84	0.97	0.97	0.97	0.96

IV. EXPERIMENTAL EVALUATION

Datasets. We use two real-world datasets for experiments: the (1) *Music* dataset[1] and (2) *Car* dataset[2]. We observed that both real-world datasets have missing and incorrect values. We report the results for the approximate band ODs $\mathsf{cat\#} \mapsto_\Delta \mathsf{year}$ over the *Music* dataset and $\mathsf{VIN} \mapsto_\Delta \mathsf{year}$ over the *Car* dataset.

Real-world Datasets. We categorize the real-world data into two groups by sampling the datasets.

- *Music-Real* is a random sample of the *Music* dataset.
- *Car-Real* is a random sample of the *Car* dataset.

CER Datasets. Although the real-world *Music* and *Car* datasets have real errors, we also randomly modify both datasets with synthetic errors to control the error rate and denote them as CER datasets.

Gold Standard. We verify the ground truth as follows.

[2]www.classicdriver.com

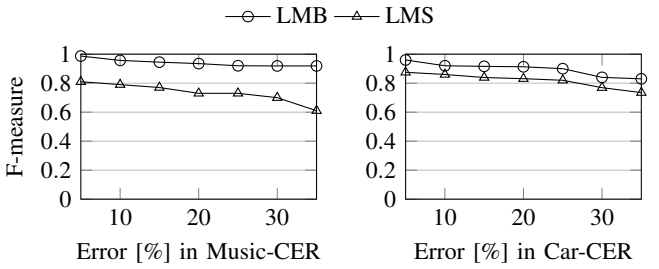

Fig. 2. Discovery quality on *Music* and *Car* datasets.

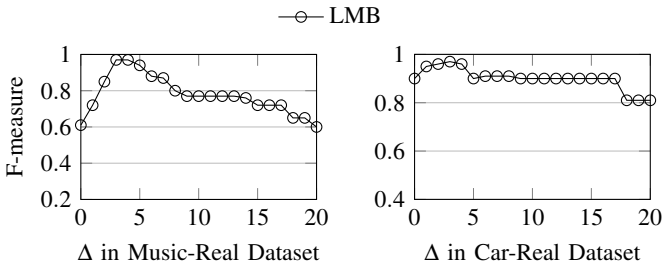

Fig. 3. Discovery when varying band-width Δ.

- *Real-world Datasets*: For real-world datasets, we manually verify the correctness of outliers wrt abODs of all variations.
- *CER datasets*: We use manually-verified ground truth of outliers wrt abODs over real-world datasets for CER-datasets.

Algorithms. We compare our LMB algorithm with LMS that discovers abODs by the concept of longest monotonic subsequences (LMS) [2]. It can detect erroneous values in each input sequence, but does not allow small variations.

Measurements: We compare in our experiments tuples discovered as outliers with the gold standard and measure the result by *precision* (P), *recall* (R) and *F-measure*(F), where $P = \frac{tp}{tp+fp}$, $R = \frac{tp}{tp+fn}$, and $F = \frac{2PR}{P+R}$. We count *tp*, *fp* and *fn* as the number of true positive, false positive and false negative tuples, respectively.

Quality of abOD Discovery. Table II presents the results of the approximate band OD discovery on the real-world datasets. We observe that LMS achieves high recall with a large loss in precision, as it does not handle small violations to monotonicity (treating them as outliers). Our proposed algorithm LMB overcomes the problem of LMS. It dominates the baseline approach (F-measure above 0.97 and improved by up to 37%) and also achieves high precision and recall over both datasets.

Figure 2 illustrates the quality results of approximate band OD discovery on the *Music* and *Car* CER datasets. We observe analogous behaviors of both LMS and LMB algorithms with the controlled error rate as in the real datasets. Our proposed algorithm LMB achieves high F-measure (above 0.9) for a reasonable amount of noise (up to 25%).

Band-Width Variations. For LMB algorithm, we vary the band-width parameter to evaluate the effect of this parameter (note that LMS is a special case of LMB where Δ is equal to 0). Our solution achieves the best F-measure when $\Delta = 3$, shown in Figure 3. This is because, in *music* industry, year denotes release date of the records, cat# is assigned to a record at early stages of the production, and the lifespan of producing music records varies from a short period of time up to a few years based on the complexity of the product and available resources. We have similar observations for car production in automobile industry.

We also observe that the precision of LMB tends to decrease when the band-width increases (not shown in Figure 3). This is because as Δ increases, the algorithm is more tolerant to monotonicity violations, which leads to lower precision and thus higher recall.

V. Conclusions

We introduce the notion of approximate band order dependency to model real-world scenarios where slight violations to ODs among attributes can be observed. We also provide effective algorithms for the approximate band OD discovery problem, which shows its soundness in theory and empirical evaluations.

In future work, we plan to consider cases where approximate band OD dependencies are only conditionally satisfied over the subsets of the data with the mix of ascending and descending orders. We will improve the efficiency and automation degree of our approximate band OD discovery solutions. We plan to adapt sampling techniques used for functional dependency and key discovery [11] and utilize distributed computing as in previous work on data discovery that includes order operators for ODs [7] to further improve the efficiency of our discovery algorithm.

References

[1] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "TANE: an efficient algorithm for discovering functional and approximate dependencies," *Comput. J.*, vol. 42, no. 2, pp. 100–111, 1999.

[2] L. Golab, H. Karloff, F. Korn, A. Saha, and D. Srivastava, "Sequential dependencies," *PVLDB*, vol. 2, no. 1, pp. 574–585, 2009.

[3] P. Langer and F. Naumann, "Efficient order dependency detection," *The VLDB Journal*, vol. 25, no. 2, pp. 223–241, 2016.

[4] J. Szlichta, P. Godfrey, and J. Gryz, "Fundamentals of order dependencies," *PVLDB*, vol. 5, no. 11, pp. 1220–1231, 2012.

[5] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava, "Effective and complete discovery of order dependencies via set-based axiomatization," *PVLDB*, vol. 10, no. 7, pp. 721–732, 2017.

[6] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava, "Effective and complete discovery of bidirectional order dependencies via set-based axiomatization," *VLDB Journal*, vol. 24, no. 7, pp. 573–591, 2018.

[7] H. Saxena, L. Golab, and I. Ilyas, "Distributed dependency discovery," *PVLDB*, vol. 12, no. 11, pp. 1624–1636, 2019.

[8] S. Ginsburg and R. Hull, "Order dependency in the relational model," *Theoretical Computer Science*, vol. 26, no. 1-2, pp. 149 – 195, 1983.

[9] J. Szlichta, P. Godfrey, J. Gryz, and C. Zuzarte, "Expressiveness and complexity of order dependencies," *PVLDB*, vol. 6, no. 14, pp. 1858–1869, 2013.

[10] X. Chu, I. Ilyas, and P. Papotti, "Discovering Denial Constraints," *PVLDB*, vol. 6, no. 13, pp. 1498–1509, 2013.

[11] T. Papenbrock and F. Naumann, "A Hybrid Approach to Functional Dependency Discovery," in *SIGMOD*, 2016, pp. 821–833.